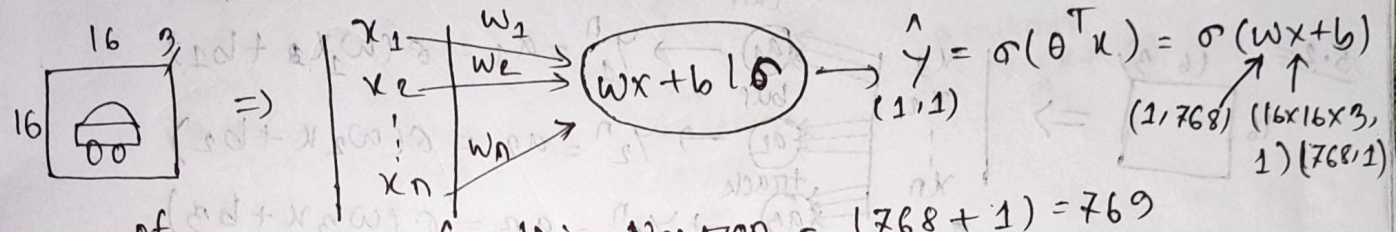
ML (1st cls after mid) D:- 12/04/23

Neuron = linear + activation
model = architecture + parameters.
Logistic Regression :- [Sigmoid function (Activation)]

* Goal-1:- Find cat's in image [1 = Present, 0 = Absent]

16×16×3 = 768



No. of parameters for this Neuron = (768 + 1) = 769

∴ so 769 Neuron needed.

* Every neuron has a bias (b).
* Every edge has a weight (w)

[Loss minimization using optimizing gradient decent algo here]

Training process :-

① Initialize weights (w) and bias (b).
② Find optimal w and b (using loss function)
③ Use $\hat{y} = \sigma(wx+b)$ to predict.

Loss Function:-

$$\mathcal{L} = -[y \log \hat{y} + (1-y) \log (1-\hat{y})] \rightarrow 1m$$

For minimization:-

$$w = w - \alpha \frac{\partial \mathcal{L}}{\partial w}$$
$$b = b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$
$\Big\} \rightarrow 1m$

[For multiple vehicle detection, Neuron should be calculated in: p×n, where p is the no. parameter in Neuron and n is the no of Neuron.
Ex: 769×3 [for 3 vehicle detection].
No. of Neuron will be increased equally if classes increase.]

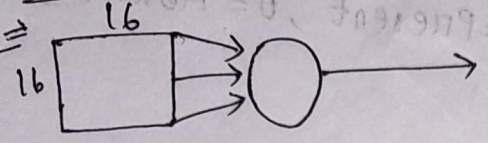
ML (end) 16.04.23

Neural Network :-

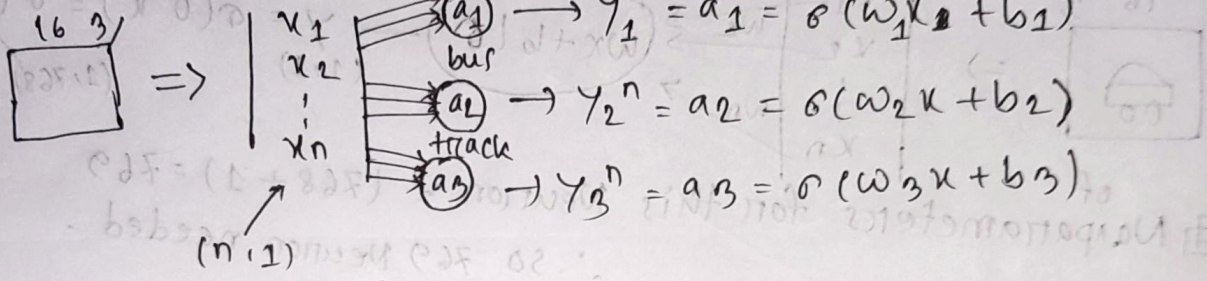
Goal-1 :- find cars in images.

goal-2 :- find car / bus / track in images

5.1 ⇒



5.2 ⇒



So, $\wedge$ → dimension of y

y → shape (3, 1)

if we insert bus as a input then output will be:

bus, $y_b = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ [column wise]

track, $y_t = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

car, $y_c = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

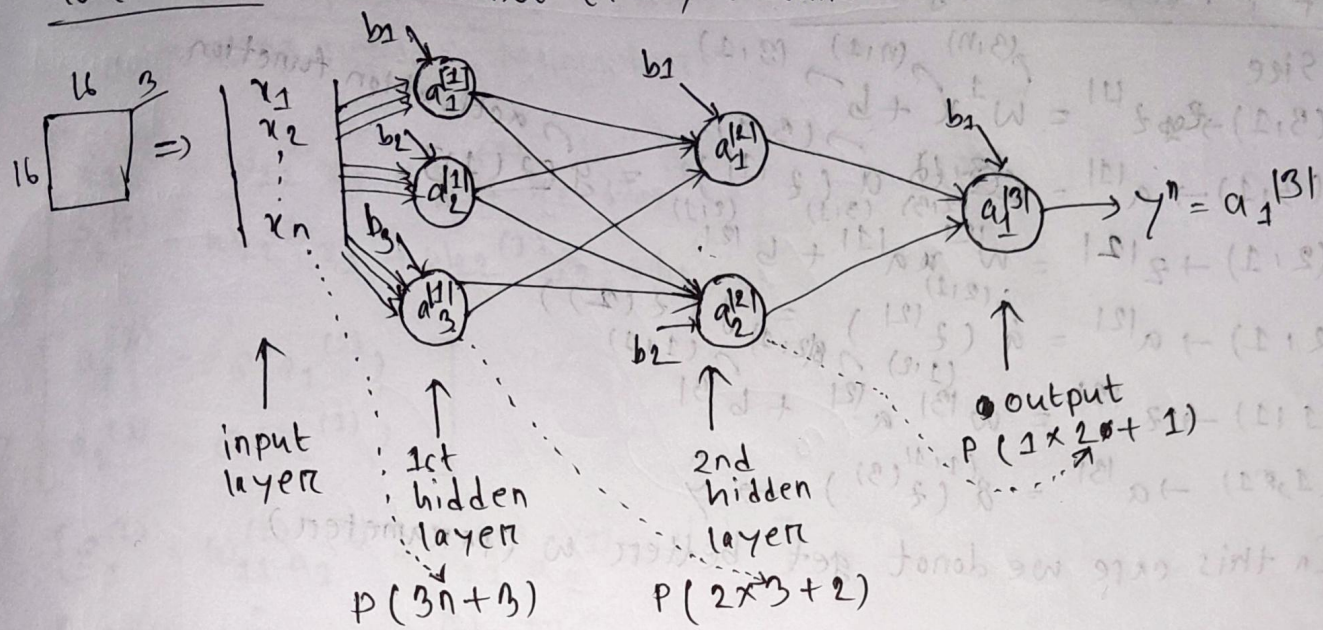
* Sigmoid doesnot take multiple output but softmax takes.
* Sometimes extra bias added to the Neuron but bias will count only 1.
* parameter for this function ; $3n+3$

L.f :-

$$\mathcal{L} = -\sum_{k=1} [y_k \log \hat{y}_k + (1-y_k) \log (1-\hat{y}_k)]$$

If single input then loss function,
If multiple input then cost function.

Goal – 3 :- Can or not [Fully connected network]



Now,

total parameters = [(3n+3) + (2×3+2) + (1×2+1)]

= [3n+3+8+3]

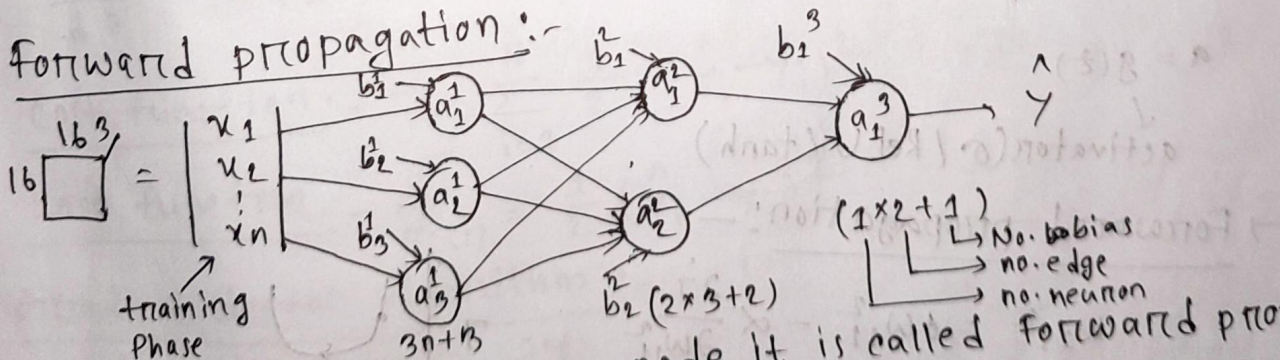
= [3n + 14] A .

ML (3rd) online

04.05.23

Neuron = Linear + Activation
(straight line) (wx+b) (sigmoid, ReLU etc)

forward propagation :-



when we get value in every node it is called forward propagation.

$a_1^1 = \sigma(z) = \sigma(wx+b)$

F.P : (left to right) :—

Size

$(3,1) \rightarrow z^{[1]} = w^{1}x + b$ (w: $(3,n)$, x: $(n,1)$, b: $(3,1)$)

$(3,1) \rightarrow a^{[1]} = a(z^{[1]}) = g[z(1)]$ (a: $(3,1)$; g → activation function)

$(2,1) \rightarrow z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$ (w: $(2,3)$, a: $(3,1)$, b: $(2,1)$)

$(2,1) \rightarrow a^{[2]} = a(z^{[2]}) = g(z(2))$ (a: $(2,1)$)

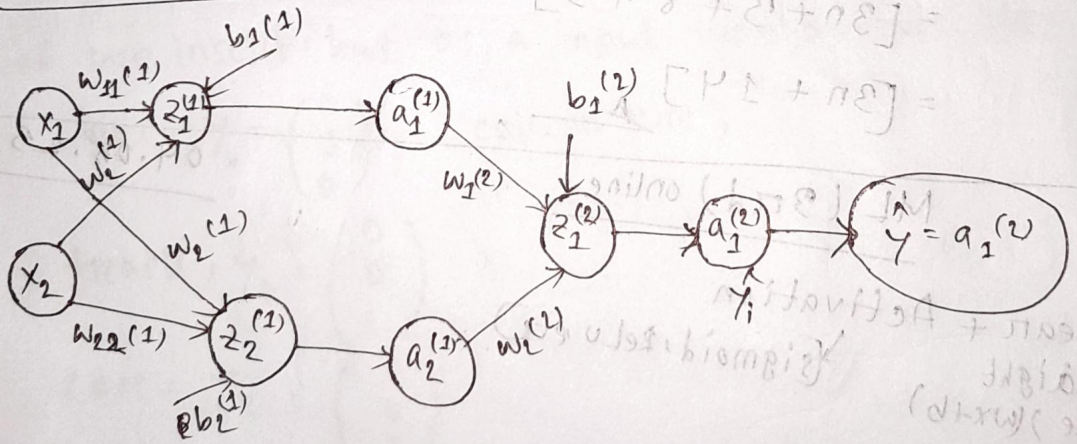
$(1,1) \rightarrow z^{[3]} = w^{[3]} a^{[2]} + b^{[3]}$ (w: $(1,2)$, a: $(2,1)$, b: $(1,1)$)

$(1,1) \rightarrow a^{[3]} = a(z^{(3)}) \rightarrow \hat{y}$ (a: $(1,1)$)

*In this case we donot get better w (parameter)

ML (4th) 07.05.23

Neural Networks :—



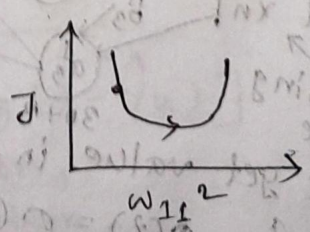
$a = g(z)$

↓

activator (σ / ReLU / tanh)

→ Forward propagation :—

$w_{11}^{2} = w_{11}^{2} - \alpha \frac{\partial J}{\partial w_{11}^{2}}$



→ **Back propagation :-** (aims to minimize the cost function by adjusting network parameters).

$$\begin{bmatrix} z_1^{(1)} = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + b_1^{(1)} \\ z_2^{(1)} = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + b_2^{(1)} \end{bmatrix}$$

$$\begin{bmatrix} a_1^{(1)} = \sigma(z_1^{(1)}) \\ a_2^{(1)} = \sigma(z_2^{(1)}) \end{bmatrix}$$

$$\left[z_1^{(2)} = w_{11}^{(2)} a_1^{(1)} + w_{21}^{(2)} a_2^{(1)} + b_1^{(2)} \right]$$

$$\left[a_1^{(2)} = \hat{y}_1 = \sigma(z_1^{(2)}) \right]$$

if, $y = \sigma(z) = \frac{1}{1+e^{-z}}$

then, $\frac{dy}{dz} = (1+e^{-z})^{-2} e^{-z}$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} \left[\frac{1}{1+e^{-z}} \left(\frac{1+e^{-z}-1}{1+e^{-z}} \right) \right.$$

$$\left. = y\left(1 - \frac{1}{1+e^{-z}}\right) \right]$$

$$= y(1-y)$$

again,

$a = \sigma(z)$

$\frac{da}{dz} = a(1-a)$ [same as $\frac{dy}{dz}$]

cost function :- $J = \sum_{i=1}^{N} \frac{1}{2} (\hat{y}_i - y_i)^2$

Loss function :- $\mathcal{L}_{(i)} = \frac{1}{2} (\hat{y}_i - y_i)^2$

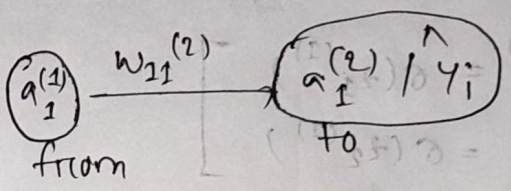
Gradient decent algorithm :-

$$w_{11}(2) = w_{11}^{(2)} - \alpha \frac{\partial \mathcal{L}_i}{\partial w_{11}(2)}$$

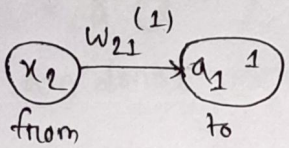
chain rule :—

$$\frac{\partial \mathcal{L}_i}{\partial w_{11}^{(2)}} = \underbrace{\frac{\partial \mathcal{L}_i}{\partial \hat{y}_i}}_{(\hat{y}_i - y_i)} \cdot \underbrace{\frac{\partial \hat{y}_i}{\partial z_1^{(2)}}}_{\hat{y}_i(1-\hat{y}_i)} \cdot \underbrace{\frac{\partial z_1^{(2)}}{\partial w_{11}^{(2)}}}_{a_1^{(1)}}$$

$$= a_1^{(1)} \partial \hat{y}_i \cdot$$



$\frac{\partial \mathcal{L}_i}{\partial w_{11}^{(2)}}$:



$$= \frac{\partial \mathcal{L}_i}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_i^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} \cdot \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} \cdot \frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}}$$

$$(\hat{y}_i - y_i) \quad \hat{y}_i(1-\hat{y}_i) \quad w_{11}^{(2)} \quad a_1^{(1)}(1-a_1^{(1)}) \quad x_2$$

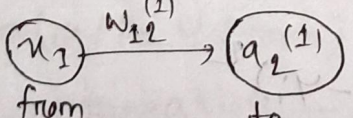
$$= x_2 a_1^{(1)} (1 - a_1^{(1)}) w_{11}^{(2)} \partial \hat{y}_i$$

$$= x_2 a_1^{(1)} (1 - a_1^{(1)}) \delta a_1^{(1)} \qquad [w_{11}^{(2)} \cdot \delta \hat{y}_i \rightarrow \delta a_1^{(1)}]$$

from $t_0(1-t_0)$; δt_0 ; $[a_1^{(1)} = t_0]$

$$\delta a_1^{(1)} \rightarrow w_{11}^{(2)} \delta \hat{y}_i \rightarrow (\hat{y}_i - y_i) \cdot \hat{y}_i (1 - \hat{y}_i)$$

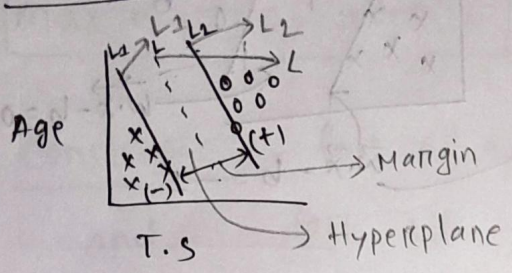
$\frac{\delta \mathcal{L}_i}{\partial w_{12}^{(1)}}$:



$$= x_1 a_2^{11} (1 - a_2^{(1)}) \underbrace{\delta a_2^{(1)}}_{w_{21}^{(2)} \delta \hat{y}_i}$$

ML (5th) 10.05.23

Support vector Machine (SVM) :- (is a classifier)

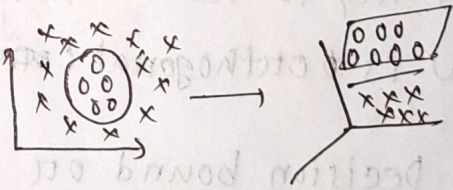


this one is hard margin SVM because target dataset is in the margin

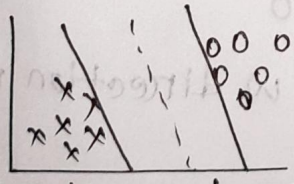
- SVM target maximum margin as seperator.
- Hyperplane's target is to increase margin.
- Logistic Regression and SVM are almost same except:-
  - L.R follows probabilistic approach and SVM follows statistical approach.
- Points we use for linear line is called support vector (points at the margin).
- SVM is an max margin classifier.

Types of SVM :-

(i) Linear SVM
(ii) Non-linear SVM

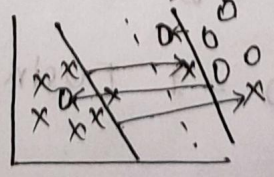


1. Hard Margin SVM :-



When there is no datapoint inside margin. (In real life this kind of SVM is hard to find)

2. Soft Margin SVM :-

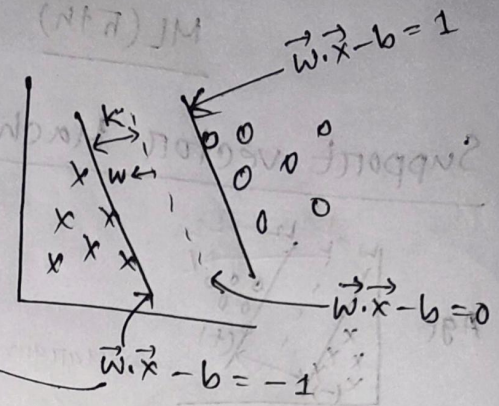


When there is same or different datapoint inside margin. (This is an real life scenario).

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad (Y = +1, -1)$$

data = n

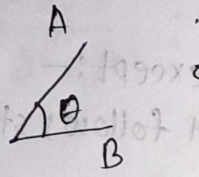
features = m



equation for hyperplane

$$\vec{w}.\vec{x} - b = 0 \quad \text{——} (1)$$

$$\Rightarrow w_1 x_1 + w_2 x_2 + \cdots + w_m x_m - b = 0$$ [every dimension will get a w]



(for w)

$$A \cdot B = |A||B| \cos\theta$$
$$= |A| \cos\theta |B|$$

$$A \cdot B = A_1 \times B_1 + A_2 \times B_2 + \cdots$$

Size of margin :- [Maximizing margin is the target of SVM].

we will try to reach margin by following the path of w. [w is orthogonal (~~mp~~ ~~mg~~) base with the middle line].

x is on Decision bound or Hyperplane, then,

$$\vec{w}.\vec{x} - b = 0 \quad \text{——} (1)$$

$$\Rightarrow [w_1 x_1 + w_2 x_2 + \cdots + w_m x_m - b] = 0$$

After ~~moding~~ moving k unit towards w direction we get,

$\vec{w}$ ← unit vector

$$\vec{w}.\left(\vec{x} + k\frac{\vec{w}}{|w|}\right) - b = 1$$

$$\Rightarrow \vec{w}.\vec{x} + k.\frac{\vec{w}.\vec{w}}{|w|} - b = 1$$

$$\Rightarrow k\frac{|w||w|\cos 0^\circ}{|w|} + 0 = 1 \quad [\text{using equ} - i]$$

$$\Rightarrow k = \frac{1}{|w|}$$

Margin size = $2k = \frac{2}{|w|}$ [if w is smaller, then k is bigger or margin is bigger]

max margin = min|w|

Constrains:- for all $y_i = 1$, $\vec{w} \cdot \vec{x}_i - b \geq 1$

and for all $y_i = -1$, $\vec{w} \cdot \vec{x}_i - b \leq -1$

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$$

So in hard margin SVM ⟶

* $\underset{w,b}{\text{argmax}} \frac{2}{|w|}$, such that $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$

or,

* $\underset{w,b}{\text{argmin}} |w|$, such that $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$.

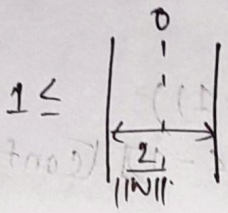
support vector, $y_i(\vec{w} \cdot \vec{x}_\phi - b) = 1$

SVM :-

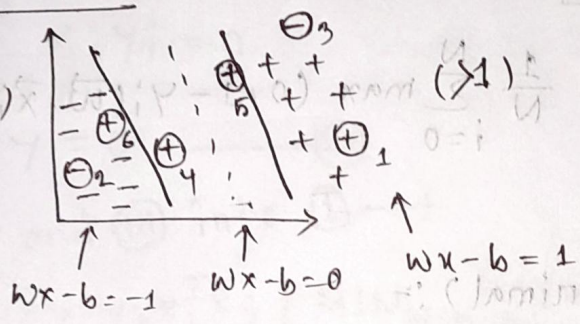
$\text{argmin}_{w,b} \|w\|$, such that $y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1 \; \forall i = 1, \ldots, N$

(y_i: true, $\vec{w}\cdot\vec{x}_i - b$: prediction)

$\max \frac{2}{\|w\|}$ ($\frac{2}{\|w\|}$: margin)



Soft margin SVM :-



Hinge loss :- (For soft SVM to calculate maximum loss of data according to class).

$$HL = \max(0, 1 - y_i(\vec{w}\cdot\vec{x}_i - b))$$

according to point we put this

As $(1 - 1)$ is negetive and 0 is bigger so max is 0.

1st point :-

$HL = \max(0, 1 - 1\ (>1))$ ($w \cdot x_i - b > 1$)

$= 0$

2nd point :-

$HL = \max(0, 1 - (-1)\ (<-1))$

$= 0$

3rd point :-

$HL = \max(0, 1 - (-1)\ (>1))$

$= >1$ or 0.

4th point :-

$HL = \max(0, 1 - (-1)$ (between 0 and 1))

= between 0 and 1.

5th point :-

HL = max (0, 1 − 1 (between 0 and 1))
= between 0 and 1.

6th point :-

HL = max (0, 1 − 1 (< −1))
=> 1. [or 0 and < −1] [confused]*

Soft margin SVM :-

$$= \text{argmin}_{w,b} \frac{1}{N} \sum_{i=0}^{N} \max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)) + \lambda \|w\|^2$$

SVM Dual :-

→ Original form (Primal) :-

$\text{MIN}_{w,b} \frac{1}{2}\|w\|^2$, such that $[\|w\|^2 = w \cdot w = w^T \cdot w]$

$y_i(\underbrace{w \cdot x_i}_{w^T \cdot x_i} - b) \geq 1, \quad \forall i = 1, 2, \ldots, N$

$\|w\| \triangleq \frac{1}{2}\|w\|^2$

Lagrangian :-

$\frac{1}{2}\|w\|^2$

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i (y_i(w^T x_i - b) - 1), \quad \text{---- (1)}$$

where α_i = lagrange multiplier.

[Original format will use Lagrian formet to create dual formet].

SVM Dual :-

$$\text{MAX}_{\alpha_i \geq 0} [\text{MIN}_{w,b} \mathcal{L}(w, b, \alpha)]$$

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

$$\Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \quad \text{——— (II)}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0, \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\Rightarrow \alpha^T y = 0 \quad \text{——— (III)}$$

satisfying (II) and (III) into (I) ⟶

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \; [\text{MIN}_{w,b} \mathcal{L}(w, b, \alpha)]$$

From SVM Dual :-

$$\underset{\alpha_i \geq 0}{\text{Max}} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \right]$$

for datapoints those are not support vector, $\alpha_i = 0$.

~~$$\underset{\alpha_i \geq 0}{\text{MIN}} \left[\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum \alpha_i \right]$$~~

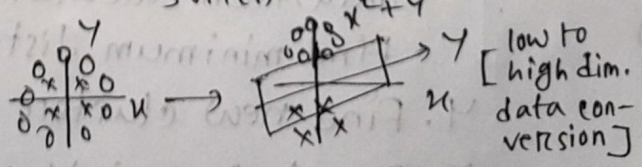
~~(Pair of SVM)~~ SVM(i)

$$\underset{\alpha_i \geq 0}{\text{MIN}} \left[\frac{1}{2} \sum_{\text{pair of SVM}(i,j)} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{\text{SVM}(i)} \alpha_i \right]$$

Why Dual ? :-

⟶ kernal friendly.

$x \rightarrow T(x)$ [dual formet of for kernal]

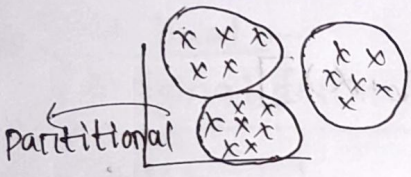


7th (ML) 17-05-23

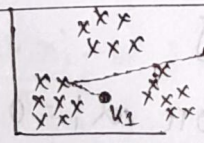
Unsupervised learning :-

Clustering :- (Grouping unlebeled examples)

- Clustering
  - (Generates tree) Hierarrical
    - Agglomerative (Bottom up)
    - Divisive (Top-down)
  - partitional
    - k-means (No-overlaping)
    - Fuzzy c-means (overlaped classes)



k-means clustering :-



$\rightarrow \sqrt{(x_n - c_j)^2}$ = Euclidian distance.

data, centroid

Steps :-

1. Selects value of k = 2 (as example).
2. Selects k-random points (centroids).
3. Assign each datapoints to nearest centroids. (selects a data, then calculates distance from the data to k and find minimum distance).
4. Find new cluster center by taking average.
5. Repeat 3 and 4 until none of the cluster center changes.

~~Repeat all level until~~ .

Selection of k :- (Elbow method)

k = 2, 3, 4, ---- 10

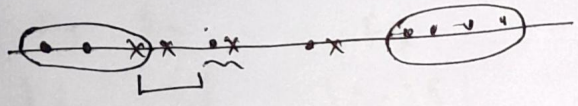
↑ varience : sum of squeries.

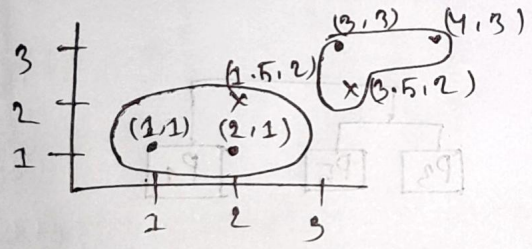
sum of squares :-

$$SS = \sum_{i=1}^{n} (x_n - c_j)^2$$ where, x_n = data points,

c_j = center of the cluster in which data points belongs to

*Sometimes this may can classify classes properly but choose data in a wrong ~~cla~~ another class then it creates a problem. we can solve that by taking value of k again in another points (value).





Update:

$$k_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right)$$

$$= (1.5, 1)$$

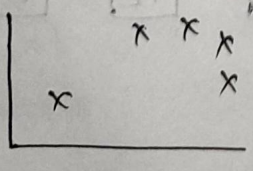
$$k_2 = \left(\frac{3+4}{2}, \frac{3+3}{2}\right)$$

$$= (3.5, 3)$$

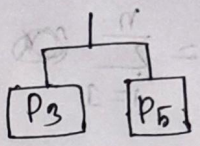
[this is the way where the value of ak will be taken according to average data points).

Hiarical clusterring :-

Euclidian distance.

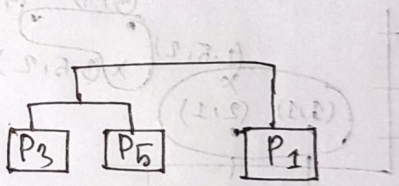


	P_1	P_2	P_3	P_4	P_5
P_1	0	9	3	6	11
P_2	9	0	7	5	10
P_3	3	7	0	9	2
P_4	6	5	9	0	8
P_5	11	10	2	8	0



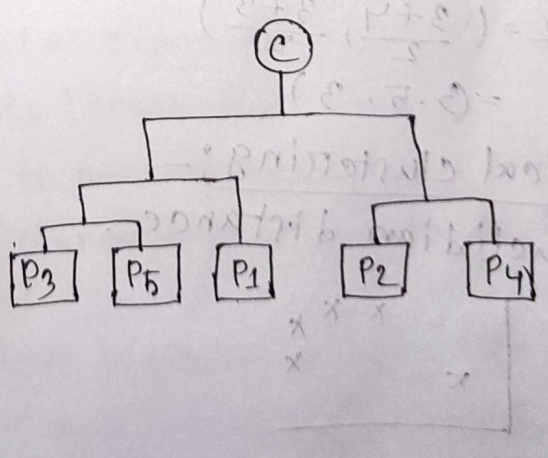
$(P_3, P_5) = 2$ (min)

	P_1	P_2	P_3, P_5	P_4
P_1	0	9	3	6
P_2	9	0	7	5
P_3, P_5	3	7 (min(7,10))	0	8
P_4	6	5	8 min(9,8)	0



$(P_1, P_3, P_5) = 3$

	P_1, P_3, P_5	P_2	P_4
P_1, P_3, P_5	0	7	6
P_2	min(9,7) 7	0	5
P_4	min(6,8) 6	5	0



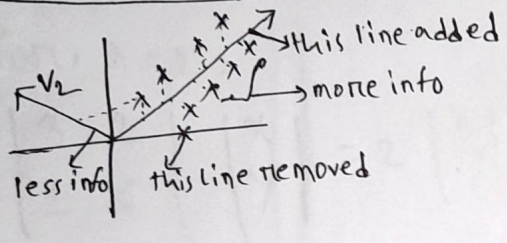
Dimensity Reduction :-

Convert higher dimensity data to lower dimensity data.

Motivation :-

→ Data compression
→ Visualization.

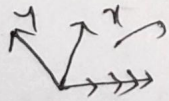
PCA (Principle Component analysis) :- (for dimensity reduction)



f_1	f_2		f_3
2	3	$\Rightarrow$	2.15
5	9		5.5
6	8		6.9

* less variation in data :-
  ↳ more information.

Eigen vector :-



$\lambda \rightarrow$ (Eigen value) $\rightarrow$ more lembda more information.

PCA :-

Eigen vector :- Eigenvector of a square matrix (A) is defined as a non-zero vector (V) by which, when a given matrix (A) is multiplied, it is equal to a scaler multiplier (λ) of that vector.

$A.V = \lambda.V$, A = given matrix
V = Eigen vector
λ = Eigen value.

Ex :-

$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$, $V = 2$, $\lambda = ?$

$\Rightarrow A \cdot V = \lambda \cdot V$

$\Rightarrow AV - \lambda V = 0$

$\Rightarrow AV - I\lambda V = 0$ [$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ = identity matrix]

$\Rightarrow V(A - I\lambda) = 0$

As, $V \neq 0$, $A - I\lambda = 0$

Taking determiniant,

$\det(A - I\lambda) = 0$

$\Rightarrow \det\left(\begin{vmatrix} 3 & 2 \\ 2 & 6 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} \right) = 0$

$\Rightarrow \det \begin{vmatrix} 3-\lambda & 2 \\ 2 & 6-\lambda \end{vmatrix} = 0$

$\Rightarrow (3-\lambda)(6-\lambda) - 4 = 0$

$\Rightarrow \lambda^2 - 9\lambda + 14 = 0$

$\therefore \lambda = \underline{7}, 2$

↑ Primary eigenvalue (as max value)

For, $\lambda = 7$,

$\begin{vmatrix} 3 & 2 \\ 2 & 6 \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix} = 7 \begin{vmatrix} x \\ y \end{vmatrix}$ $(AV = \lambda V)$

$\Rightarrow 3x + 2y = 7x$ —— (i)

again,

$2x + 6y = 7y$ —— (ii)

by solving (i) and (ii)

$\therefore x = 1, y = 2$

$V = \begin{vmatrix} 1 \\ 2 \end{vmatrix}$, $\lambda = 7$

Unit vector :- Any vector can be a unit vector by dividing it by magnitude of the vector.

$$|V| = \sqrt{1^2+2^2}$$
$$= \sqrt{5}$$

so, unit vector of $v = \begin{vmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{vmatrix}$ $\left(v = \frac{\vec{v}}{|V|}\right)$

for, $\lambda = 2$,

$$\begin{vmatrix} 3 & 2 \\ 2 & 6 \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix} = 2 \begin{vmatrix} x \\ y \end{vmatrix}$$

$\Rightarrow 3x + 2y = 2x$ —— ①

again,

$2x + 6y = 2y$ —— ⑪

by solving ① and ⑪ we get,

$x = 2, y = 1$

$$v = \begin{vmatrix} 2 \\ 1 \end{vmatrix}$$

unit vector $= \begin{vmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{vmatrix}$

H.W: $A = \begin{vmatrix} 6 & 2 \\ 2 & 6 \end{vmatrix}$

PCA :-

Given data, $M = \begin{vmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{vmatrix}$ ← convert it to 1-D

$$M^T = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{bmatrix}$$

co-variance matrix, θ [co-variance matrix is an square matrix]

→ Eigenvector

$$A = M^T M = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{vmatrix} \times \begin{vmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{vmatrix}$$

$2\times 7 \rightarrow 7\times 2$
$\Rightarrow 2\times 2$

$$= \begin{vmatrix} 30 & 28 \\ 28 & 30 \end{vmatrix}$$

Taking determinant,

$$\begin{vmatrix} 30 & 28 \\ 28 & 30 \end{vmatrix} V - I\lambda V = 0$$

$$\Rightarrow \begin{vmatrix} 30 & 28 \\ 28 & 30 \end{vmatrix} - I\lambda = 0 \quad [A - I\lambda = 0]$$

$$\Rightarrow \det\left(\begin{vmatrix} 30 & 28 \\ 28 & 30 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} \right) = 0$$

$$\Rightarrow (30-\lambda)^2 = 28^2$$

$$\Rightarrow 30 - \lambda = \pm 28$$

$$\Rightarrow \lambda = 58 \text{ and } 2$$

P.E.V

using $\lambda = 58$,

$AV = \lambda V$

$$\Rightarrow \begin{vmatrix} 30 & 28 \\ 28 & 30 \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix} = 58 \begin{vmatrix} x \\ y \end{vmatrix}$$

$$\Rightarrow 30x + 28y = 58x \quad \text{---(I)}$$

Again,

$$28x + 30y = 58y \quad \text{---(II)}$$

$x = 1, y = 1$

$$v = \begin{vmatrix} 1 \\ 1 \end{vmatrix}$$

unit vector, $v = \begin{vmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{vmatrix}$

Data compression,

New data points,

= M x unit vector,

$$= \begin{vmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{vmatrix} \times \begin{vmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{vmatrix}$$

$$= \begin{vmatrix} 2.12 \\ 2.12 \\ 4.94 \\ 4.94 \end{vmatrix}$$

steps: covarient matrix → Eigen value → Eigen vector → unit vector → Data compression.

ML (9th) 24-05-23

Recomender system :- (utility matrix)

① content based (based on your search)

② collabrate filtering (based on your friends similar interest)

collabrate filtering :-

→ utility matrix

→ Measure (distance utility)

↳ Jaccard distance

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{A \cap B}{|A| + |B| - |A \cap B|}$$

(need more distance)

↳ cosine similarity

$$sim(A,B) = \cos\theta = \frac{A \cdot B}{|A|\,|B|}$$

(θ need to be less)

→ Normalization.

customer \ item	A	B	C
JOHN	5	3	2
MARK	3	4	?
JUCY	?	2	5

Avg. difference in ratings :-

Item(A,B) = ((5 - 3) + (3 - 4)) / 2

= 0.5

Item (A, C) = (5-2) / 1 = 3

Item (B, C) = ((3-2) + (2-5)) / 2
= -1

Rating of JUCY(A) :-

Rating of A item in term of B = 2 + 0.5 = 2.5

(2 ↑ rating of JUCY(B); 0.5 ↑ Avg dif (A, B))

Rating of A in terms of C = 5 + 3 = 8

Average normalized rating = ~~2+5~~ $\frac{(2.5 \times 2) + (8 \times 1)}{3}$

= 4.33

Rating of Mark(C) :-

Rating of C in terms of A = 3 + 3 = 6

Rating of C in terms of B = 4 + (-1)
= 3

Average normalized rating = $\frac{(6 \times 1) + (3 \times 2)}{3}$

= 4

So, updated,

i / C	A	B	C
John	5	3	2
Mark	3	4	4
Jucy	4.33	2	5

cosine similarity :-

customer \ item	1	2	3	4	5	6	7	8
U_1	5	1	4	?	?		2	1
U_2	5		3	(2)	5		2	1
U_3	1	4	2	(5)	2		5	4

cosine similarity,

$$sim(U_1, U_2) = \frac{U_1 \cdot U_2}{|U_1| \cdot |U_2|} = \cos\theta$$

U_1	5	4	2	1
U_2	5	3	2	1

$$sim(U_1, U_2) = \frac{25 + 12 + 4 + 1}{\sqrt{5^2 + 4^2 + 2^2 + 1^2} \cdot \sqrt{5^2 + 3^2 + 2^2 + 1^2}}$$

$$= 0.99$$

U_1	5	1	4	2	1
U_3	1	4	2	5	4

$$sim(U_1, U_3) = 0.57$$

$$R(U_1, 4) = \frac{sim(U_1, U_2) \times 2 + sim(U_1, U_3) \times 5}{sim(U_1, U_2) + sim(U_1, U_3)}$$

$$= 3.1$$